Theoretical Computer Science 81 (1991) 295-304
Elsevier

Recognizing different types of beta-cycles in a database scheme*

To-Yat Cheung

Department of Computer Science, University of Ottawa, Ottawa, Ontario, K1N 9B4, Canada

Yunzhou Zhu

Department of Computer Science, University of Science and Technology of China, Hefei, Anhui, China

Communicated by G. Ausiello
Received April 1987
Revised November 1988

Abstract

Cheung, T.-Y., and Y. Zhu, Recognizing different types of beta-cycles in a database scheme, Theoretical Computer Science 81 (1991) 295-304.

This paper first distinguishes between three types of beta-cycles, namely pure beta-cycles, beta-triangles and non-pure beta-cycles. An algorithm is then proposed for detecting their presence or absence in a database scheme. The database scheme is beta-acyclic if all of them are absent. The complexity of the algorithm is $O(|N||E|^2)$, where $|N|$ is the total number of attributes and $|E|$ is the number of non-empty pairwise intersections of the relation schemes. The algorithm may be used to obtain a beta-acyclic scheme through a transformation process that eliminates beta-cycles from a database scheme by suitably adding attributes into its relation schemes.

1. Introduction

It is well known that the design and management of a database will be significantly simplified if the database scheme satisfies an acyclicity property. For acyclic database schemes, there often exist efficient (polynomial time) algorithms for solving problems that are NP-complete for general database schemes [10]. For example, an alpha-acyclic database scheme satisfies such desirable properties [2, 6] as equivalence

* This research is supported by the Natural Sciences and Engineering Research Council of Canada under Grant No. A8963.

between join dependency and multi-valued dependency, implication of global consistency by pairwise consistency, existence of an optimal semi-join program for distributed query processing, etc.

Beeri et al. [2], Fagin [6] and Ausiello et al. [1] described many conditions for characterizing three important types of acyclicity, namely alpha-acyclicity, beta-acyclicity and gamma-acyclicity. It was proven that gamma-acyclicity $\Rightarrow$ beta-acyclicity $\Rightarrow$ alpha-acyclicity but that none of the reverse implications hold [6]. One of the fundamental problems is how to efficiently recognize these acyclicity properties in a database scheme. There exist many recognition algorithms in the literature. A linear algorithm for determining alpha-acyclicity was given by Tarjan and Yannakakis [9]. An $O(|N||V||E|)$ recognition algorithm for gamma-acyclicity was presented by Zhu [11], where $|N|$ is the number of attributes, $|V|$ is the number of relation schemes and $|E|$ is the number of non-empty pairwise intersections of the relation schemes. D'Atri et al. [5] proposed three algorithms, all of order $O(|V|^2|H|)$, for recognizing alpha-, beta- and gamma-acyclicities, where $|H|$ is the length of description of the database scheme. Fagin [6] gave a sketchy description of a method for recognizing beta-acyclicity. No formal description and complexity analysis were provided. It is our estimation that its complexity is of order $O(|N||V|^5)$.

In database management, it is sometimes desirable to recognize or to have the acyclicity properties in the subschemes of a database scheme. For example, a beta-acyclic scheme has the important property that each of its subschemes is alpha-acyclic. This is not necessarily true for other types of acyclicity.

In this paper, a recognition algorithm of order $O(|N||E|^2)$ is presented for testing beta-acyclicity and detecting three types of beta-cyclic subschemes.

This paper is organized as follows. In Section 2, the definitions of the line graph of a database scheme and duplicate-reduced subgraphs of the line graph are given. Section 3 characterizes three types of beta-cyclic database schemes in terms of the corresponding line graphs and duplicate-reduced subgraphs. Based on such characterizations, an algorithm for testing beta-acyclicity and detecting these subschemes is presented in Section 4. The complexity of the algorithm is then derived and compared with those of Fagin's and D'Atri's algorithms. It is also discussed how the algorithm may be used to transform beta-cyclic database schemes into beta-acyclic ones.

2. The line graph of a database scheme

Given a universe U of attributes, a relation scheme is a non-empty subset of U and a database scheme $\boldsymbol{R}$ is a collection of relation schemes. A database subscheme is a subset of this collection.

In the literature, database schemes are often represented as hypergraphs. Though a lot of important results about acyclic databases have been described in terms of hypergraphs, it is found that the properties and the recognition algorithm developed

in this paper can be stated more clearly in terms of a related but different formalism—the line graph. The latter is also frequently used for the description of database schemes [7, 11]. Furthermore, a line graph representation is more realistic when a recognition algorithm is implemented in a distributed environment [3].

In the following, the definitions of pure cycles, incompatible triangles and duplicate-reduced subgraphs of the line graph of a database scheme are given. They are used in the next section for characterizing three types of beta-cycles in a line graph.

Consider a graph $G=(V, E)$, where V is the set of vertices and E the set of edges.

Definition 2.1. $\mu=(v_1, e_1, v_2, e_2, \ldots, v_m, e_m, v_{m+1})$, or simply $\mu=(v_1, v_2, \ldots, v_m, v_{m+1})$ when the e's are implicitly understood, is called a *cycle* of G if

(i) $m \geq 3$;

(ii) v_i, $i=1, 2, \ldots, m$, are distinct vertices of V and $v_{m+1}=v_1$; and

(iii) e_i connects v_i and v_{i+1}, $i=1, 2, \ldots, m$.

v_i and v_{i+1} are called *neighbors* in μ. An edge in G connecting two non-neighboring vertices in μ is called a *chord* of μ. In particular, μ is called a *triangle* if $m=3$ and a *pure cycle* if it does not have any chord.

A graph $G=(V, E)$ is said to be *triangulated* if every cycle of length greater than 3 has a chord.

Triangulated graphs are also called chordal, monotone transitive or rigid circuit graphs. Rose et al. [8] gave an $O(|V|+|E|)$ algorithm for testing whether a graph is triangulated.

Definition 2.2. The *line graph* $\mathrm{LG}(\boldsymbol{R})=(V, E; L)$ of a database scheme $\boldsymbol{R}=(R_1, R_2, \ldots, R_{|V|})$ consists of a graph $G(\boldsymbol{R})=(V, E)$ and a set of labels defined as follows:

(i) $v_i \in V$ if and only if $R_i \in \boldsymbol{R}$. The label $L[v_i]$ of v_i is the set of attributes R_i.

(ii) $e=(v_i, v_j) \in E$ if and only if $v_i, v_j \in V$ and $L[e] \neq \emptyset$, where $L[e]=L[v_i] \cap L[v_j]$ is called the *chordal label* of e.

A line graph is also called a complete intersection graph or a join graph [7]. Note that an edge exists in a line graph if and only if its label is not empty.

Notation

Throughout this paper, the labels of a line graph are included within braces { }, or represented by the capital letter X. Sometimes, when there is no confusion, the label of an edge or vertex is also used to identify the edge or vertex itself. For example, $\{x\}$ may mean either the label whose attribute is x or the edge whose label is $\{x\}$.

$\mathrm{LG}(\boldsymbol{R})=(V, E; L)$ denotes the line graph and $G(\boldsymbol{R})=(V, E)$ its associated graph of a database scheme $\boldsymbol{R}$. Also, the symbols $\supseteq$ and $\supset$ mean "contains" and "properly contains", respectively.

Definition 2.3. A cycle $\mu = (v_1, v_2, \ldots, v_m, v_1)$ of G is called a *pure cycle* of LG($\boldsymbol{R}$) provided that $L[v_i] \cap L[v_j] \neq \emptyset$ if and only if v_i and v_j are neighbors in μ. In addition, if $m = 3$, it also requires $L[v_1] \cap L[v_2] \cap L[v_3] = \emptyset$.

Definition 2.4. A cycle $\mu = (v_1, e_1, v_2, e_2, \ldots, v_m, e_m, v_1)$ of G is called a *beta-cycle* of LG($\boldsymbol{R}$) $= (V, E; L)$ if and only if $\mu' = (v_1', e_1', v_2', e_2', \ldots, v_m', e_m', v_1')$ is a pure cycle of LG($\boldsymbol{R}'$), where $\boldsymbol{R}' = \{R_k' = R_k - X$ (with empty R_k' deleted)$\}_{k=1,\ldots,m}$, $v_i' = \{L[v_i] - X\}$, $e_i' = \{L[e_i] - X\}$, $1 \leq i \leq m$, and $X = L[v_1] \cap L[v_2] \cap \cdots \cap L[v_m]$.

In particular, a *beta-triangle* is a beta-cycle with $m = 3$. A *pure beta-cycle* is a beta-cycle with $m > 3$ and $X = \emptyset$. A *non-pure beta-cycle* is a beta-cycle with $m > 3$ and $X \neq \emptyset$.

It is interesting to compare the line graph and hypergraph representations of a database scheme $\boldsymbol{R}$. In a hypergraph (V, E), each vertex in V represents an attribute of $\boldsymbol{R}$ and each hyperedge in E, a non-empty subset of V, represents a relation scheme. It is obvious that the vertices and edges of a line graph representation correspond to the hyperedges and their non-empty pairwise intersections in a hypergraph representation, respectively. As a consequence, our definitions of beta-cycles and beta-acyclicity in terms of line graphs are exactly the same as those given in terms of hypergraphs [2, 6].

Definition 2.5. A triangle $\mu = (v_1, e_1, v_2, e_2, v_3, e_3, v_1)$ in LG($\boldsymbol{R}$) is said to be *incompatible* if and only if $L[e_1]$, $L[e_2]$ and $L[e_3]$ are pairwise incompatible, i.e., for any v_i and v_j, $1 \leq i, j \leq 3$, $i \neq j$, $L[e_i] \not\supseteq L[e_j]$ and $L[e_j] \not\supseteq L[e_i]$.

The following types of edges and subgraphs play a special role in our investigation of beta-cyclic database schemes.

Definition 2.6. An edge $e \in E$ is said to be a *duplicate edge* of LG($\boldsymbol{R}$) $= (V, E; L)$ if there exists at least one other edge $e' \in E$ such that $L[e] = L[e']$. For a duplicate edge e, the *duplicate-reduced subgraph* of $G(\boldsymbol{R})$ with respect to e, denoted by $G'(\boldsymbol{R}, e)$, is the graph (V', E'), where $V' = \{v' \mid v' \in V$ and $L[v'] \supset L[e]\}$ and $E' = \{e' \mid e' \in E$ and $L[e'] \supset L[e]\}$.

Example 2.7. Consider the database scheme $\boldsymbol{R} = \{xab, xbc, xcdhp, xda, hk, wykm, ymn, mpnw\}$, whose line graph LG($\boldsymbol{R}$) is shown in Fig. 1. $\{v_3, v_5, v_6, v_8, v_3\}$ is a pure beta-cycle, $\{v_6, v_7, v_8, v_6\}$ is a beta-triangle and $\{v_1, v_2, v_3, v_4, v_1\}$ is a non-pure beta-cycle. $\{(v_1, v_3), (v_2, v_4)\}$ is a set of duplicate edges with chordal label $\{x\}$.

The following observations are important for the development of the propositions in the next section.

(a) The line graph of a non-pure beta-cycle μ is a complete subgraph of LG($\boldsymbol{R}$). All its chords have the same label X and all its non-chordal edges have a label properly containing X, where X is defined in Definition 2.4.

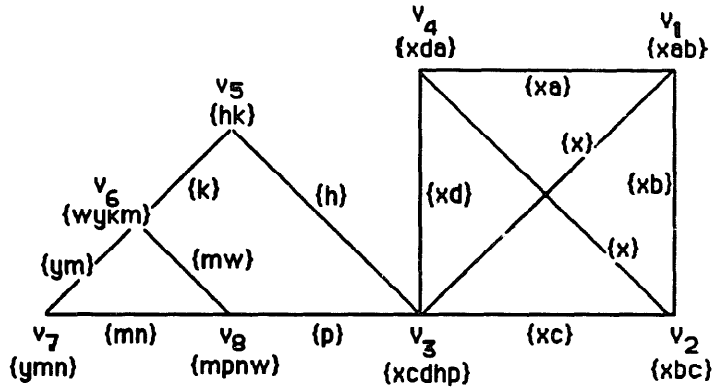

Fig. 1. The beta-triangles, pure beta-cycles and non-pure beta-cycles of the line graph of a database scheme.

This observation follows from Definition 2.4. The first part is due to the fact that, for every chord (v_i, v_j) of μ, if $L[v_i] \cap L[v_j] \neq X$, then $L[v_i'] \cap L[v_j'] = L[v_i] \cap L[v_j] - X \neq \emptyset$, contradictory to the fact that μ' is a pure cycle. The second part follows from the fact that, if $L[e_i] = X$ for a certain edge e_i of μ, then $L[e_i'] = \emptyset$, contradictory to the fact that μ' is a cycle.

(b) In creating $G'(\boldsymbol{R}, e)$ from $G(\boldsymbol{R})$, those and only those vertices and edges whose label does not properly contain $L[e]$ are removed. Such removals do not create new cycles of any kind. Also, for any non-pure beta-cycle with chordal label $L[e]$, such removals change its complete subgraph in $G(\boldsymbol{R})$ to a pure cycle in $G'(\boldsymbol{R}, e)$.

3. Characterization for three types of beta-cycles

According to Definitions 2.3 and 2.4, the beta-cycles of a line graph can be grouped into three disjoint sets: the set of pure beta-cycles, the set of beta-triangles and the set of non-pure beta-cycles.

As stated in the following three propositions, the presence or absence of each type of beta-cycles in a line graph LG($\boldsymbol{R}$) can be characterized by a condition of its associated graph $G(\boldsymbol{R})$, the line graph itself or its duplicate-reduced subgraphs $G'(\boldsymbol{R}, e)$. Proposition 3.1 follows immediately from the definitions of pure cycles and triangulated graphs, [8].

Proposition 3.1. LG($\boldsymbol{R}$) *has no pure cycles of length bigger than* 3 *if and only if* $G(\boldsymbol{R})$ *is triangulated.*

Because of Proposition 3.1, a line graph LG($\boldsymbol{R}$) is said to be triangulated if and only if its associated graph $G(\boldsymbol{R})$ is triangulated.

Proposition 3.2. *$\mu = (v_1, e_1, v_2, e_2, v_3, e_3, v_4 = v_1)$ is a beta-triangle in* LG($\boldsymbol{R}$) *if and only if it is an incompatible triangle of* LG($\boldsymbol{R}$).

Proof. Let X and the entities v_1', e_1', v_2', e_2', v_3', e_3' v_4' be defined as in Definition 2.4. It is obvious that $L[e_1'] \cap L[e_2'] \cap L[e_3'] = \emptyset$ and that $L[e_i'] = L[e_i] - X$, $i = 1, 2, 3$. It follows that μ is a beta-triangle if and only if $L[e_i'] \neq \emptyset$, $i = 1, 2, 3$.

($\Leftarrow$) Suppose, by contradiction, that μ is not a beta-triangle. Without loss of generality, we may assume that $L[e_1'] = \emptyset$. Then, $L[e_1] = L[e_1'] \cup X = X = L[e_1] \cap L[e_2] \cap L[e_3] \subseteq L[e_2]$. That is, μ is not incompatible.

($\Rightarrow$) Suppose, by contradiction, that μ is not incompatible. Assume that $L[e_2] \supseteq L[e_1]$. Then, $L[e_1] \subseteq L[e_1] \cap L[e_2] = X$ and $L[e_1'] = L[e_1] - X = \emptyset$. That is, μ is not a beta-triangle. □

Propositions 3.1 and 3.2 provide two methods for detecting pure beta-cycles and beta-triangles. However, as shown in Example 3.4, their absence in a line graph does not guarantee that the graph also has no non-pure beta-cycles. Proposition 3.3 provides a solution to this problem.

Proposition 3.3. *Let e be a duplicate edge of* LG($\boldsymbol{R}$). LG($\boldsymbol{R}$) *has no non-pure beta-cycles with chordal label $L[e]$ if and only if $G'(\boldsymbol{R}, e)$ is triangulated.*

Proof. ($\Leftarrow$) Suppose $G'(\boldsymbol{R}, e)$ is triangulated but LG($\boldsymbol{R}$) has a non-pure beta-cycle $\mu = (v_1, v_2, \ldots, v_m, v_1)$ with chordal label $L[e]$, where $m > 3$. It follows that $L[v_i] \cap L[v_j] = L[e]$ if and only if v_i and v_j are not neighbors in μ and that $L[v_i] \cap L[v_j] \supset L[e]$ if and only if v_i and v_j are neighbors. Since the removals eliminate those and only those edges whose label does not properly contain $L[e]$, all chords of μ but none of its non-chordal edges are eliminated, resulting in μ being a chordless cycle in $G'(\boldsymbol{R}, e)$ with length bigger than 3. This contradicts with the assumption that $G'(\boldsymbol{R}, e)$ is triangulated.

($\Rightarrow$) Suppose $G'(\boldsymbol{R}, e)$ is not triangulated. Then, there exists at least one chordless cycle $\mu = (v_1, e_1, v_2, e_2, \ldots, v_m, e_m, v_1)$ of $G'(\boldsymbol{R}, e)$ with length $m > 3$, where $L[v_k] \supset L[e]$ and $L[e_k] \supset L[e]$, $k = 1, 2, \ldots, m$, when v_k and e_k are considered within LG($\boldsymbol{R}$). It follows that, for any pair of non-neighboring vertices v_i and v_j in μ, $L[(v_i, v_j)] \supseteq L[e]$, $i, j = 1, 2, \ldots, m$, $i \neq j$. Hence μ has a chord (v_i, v_j) in LG($\boldsymbol{R}$). Since (v_i, v_j) has been deleted from LG($\boldsymbol{R}$), $L[e] \supseteq L[(v_i, v_j)]$. Therefore, $L[(v_i, v_j)] = L[e]$ for all pairs of non-neighboring vertices of μ. This implies that μ is a non-pure beta-cycle in LG($\boldsymbol{R}$). □

Note that, for each duplicate edge e, there may be several distinct non-pure beta-cycles. However, they all have the same duplicate-reduced subgraph $G'(\boldsymbol{R}, e)$. To detect the existence of such cycles, however, it is sufficient to check the duplicate-reduced subgraph $G'(\boldsymbol{R}, e)$ only once. Examples 3.4 and 3.5 illustrate the necessary and sufficient conditions of Proposition 3.3, respectively.

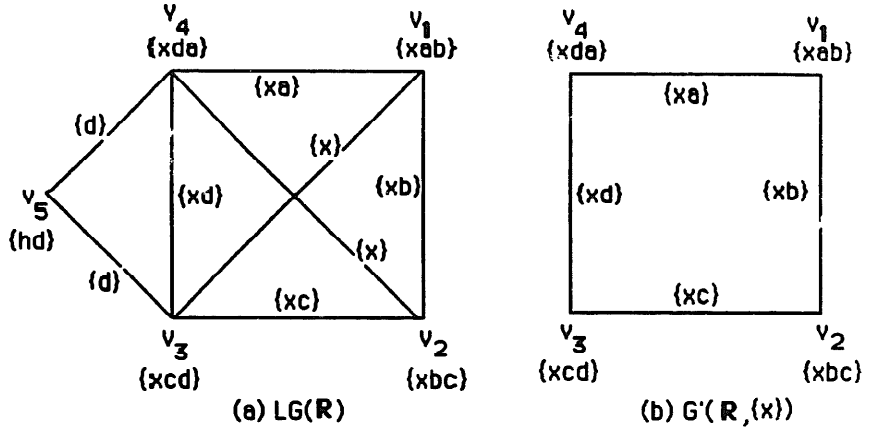

Fig. 2. (a) The line graph of a database scheme. (b) The duplicate-reduced subgraph $G'(\boldsymbol{R}, \{x\})$.

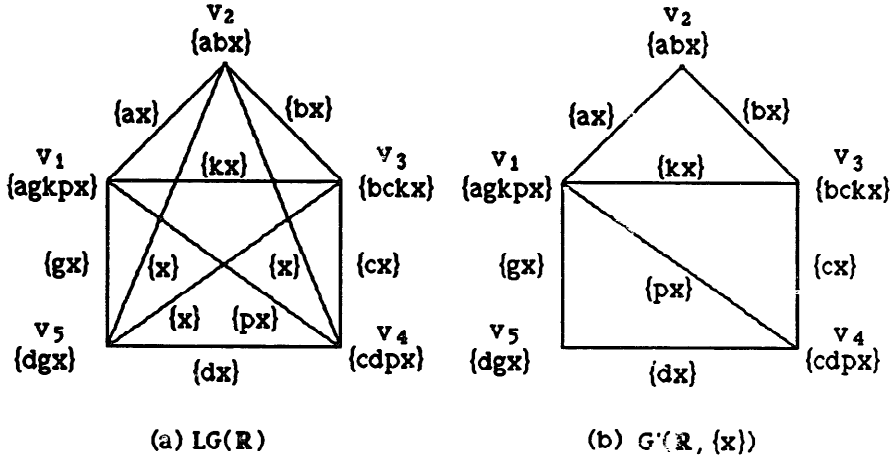

Fig. 3. LG($\boldsymbol{R}$) has no non-pure beta-cycles and $G'(\boldsymbol{R}, \{x\})$ is triangulated.

Example 3.4. Consider the database scheme $\boldsymbol{R} = \{xab, xbc, xcd, xda, hd\}$, whose line graph LG($\boldsymbol{R}$) is shown in Fig. 2a. LG($\boldsymbol{R}$) is triangulated and has no incompatible triangles, but has a non-pure beta-cycle $(v_1, v_2, v_3, v_4, v_1)$ with chordal label $\{x\}$. The duplicate-reduced subgraph $G'(\boldsymbol{R}, \{x\})$ (Fig. 2b) is not triangulated. (Note that, as a subgraph of $G(\boldsymbol{R})$, $G'(\boldsymbol{R}, \{x\})$ should have no label. The labels shown in Fig. 2b are just used to explain which edges will remain after the deletion.)

Example 3.5. Figure 3a shows the line graph of the database scheme $\boldsymbol{R} = \{agkpx, abx, bckx, cdpx, dgx\}$. Since $\{x\}$ is the only duplicate edge, any non-pure beta-cyclic subschemes (if existing) must have chordal label $\{x\}$. Since the duplicate-reduced subgraph $G'(\boldsymbol{R}, \{x\})$ (Fig. 3b) is triangulated, such beta-cyclic subschemes do not exist.

4. An algorithm for detecting beta-acyclic and beta-cyclic database schemes

According to Definition 2.4, three special types of database schemes or subschemes can be defined as follows.

Definition 4.1. A database scheme $\boldsymbol{R}$ is said to be *pure beta-cyclic, beta-triangular* or *non-pure beta-cyclic* if and only if its line graph LG($\boldsymbol{R}$) is a pure beta-cycle, a beta-triangle or a non-pure beta-cycle, respectively. $\boldsymbol{R}$ is said to be *beta-acyclic* if and only if LG($\boldsymbol{R}$) does not contain any of these beta-cyclic subschemes.

In the following, the results of Section 3 are stated as an algorithm for detecting these three types of beta-cyclic subschemes. The procedure TRIANGULATED(G) refers to Rose's algorithm [8], which accepts a graph G as input and outputs the value "true" if G is triangulated and the value "false" otherwise.

Algorithm TEST-BETA-ACYCLICITY
Input: The graph $G(\boldsymbol{R})$ and label set L of a database scheme $\boldsymbol{R}$.
Output: One of the following messages:
Message1: The database scheme has at least one pure beta-cyclic subscheme.
Message2: The database scheme has no pure beta-cyclic subschemes but has at least one beta-triangular subscheme.
Message3: The database scheme has neither pure beta-cyclic nor beta-triangular subschemes but has at least one non-pure beta-cyclic subscheme.
Message4: The database scheme is beta-acyclic.

Procedure TYPE-BETA-CYCLE($G(\boldsymbol{R})$, L): STRING;
begin
(1) **if** not TRIANGULATED($G(\boldsymbol{R})$) **then exit**(Message1);
(2) **for** each edge $(v_i, v_j) \in E$ **do**
  for each $v_k \in G(\boldsymbol{R})$, where (v_i, v_k), $(v_j, v_k) \in E$ and $k \neq i, j$, **do**
    if (v_i, v_j, v_k) is incompatible **then exit**(Message2);
(3) Determine the set of duplicate edges $\{e_1, e_2, \ldots, e_d\}$ of LG($\boldsymbol{R}$) with distinct labels;
(4) **for** $i = 1$ to d **do**
  begin
    determine the duplicate-reduced subgraph $G'(\boldsymbol{R}, e_i)$;
    if not TRIANGULATED($G'(\boldsymbol{R}, e_i)$) **then exit**(Message3);
  end
  exit(Message4);
end.

4.1. Complexity of Algorithm TEST-BETA-ACYCLICITY

For a database scheme, let $|N|$, $|V|$ and $|E|$ denote the number of attributes, the number of relation schemes and the number of pairwise intersections of the relation schemes, respectively. In this algorithm, Step (1) may be solved in time $O(|E|+|V|)$ [7, 9]. Step (2) is a double loop of order $O(|N||E||V|)$. Step (3) requires time $O(|N||E|\log|E|)$ to determine all the duplicate edges. For Step (4), it takes time $O(|N|(|E|+|V|))$ to create a duplicate-reduced subgraph and $O(|E|+|V|)$ time to

test its triangulatedness. Hence, the order of this step is $O(|N|(|E|^2+|E||V|))$. Because of its dominancy among the four steps, this term is also the order of the algorithm. Without loss of generality, we can assume that the line graph representing the database scheme is connected. Hence, $O(|V|)$ is always lower than $O(|E|)$ and $O(|N|(|E|^2+|E||V|)) = O(|N||E|^2)$.

In the following, the complexity of Algorithm TEST-BETA-ACYCLICITY is compared with those of D'Atri's [1] and Fagin's [6] algorithms. As mentioned in the introduction, D'Atri's algorithm is of order $O(|V|^2|H|)$. In particular, if $|H| = O(|N||V|)$, then its order is $O(|N||V|^3)$. Since Fagin did not clearly describe his algorithm and its complexity, we provide the following analysis. Briefly, it is claimed that a database scheme is beta-acyclic if and only if the parameter Succeeds has a "true" value after applying the following procedure.

Procedure Fagin;
Succeeds := true
  for each triplet (v_1, v_2, v_3) **do**
    if both $L[v_1] \cap L[v_2] \neq \emptyset$ **and** $L[v_2] \cap L[v_3] \neq \emptyset$ **then**
      begin
        $X := L[v_1] \cap L[v_2] \cap L[v_3]$
        $L[v_1'] := L[v_1] - X$; $L[v_2'] := L[v_2] - X$; $L[v_3'] := L[v_3] - X$
        $T := \{v \mid v = v_1$ **or** $v = v_3$ **or** $(L[v] \supset X$ **and** $L[v] \cap L[v_2'] = \emptyset)\}$
        $T' := \{v' \mid L[v'] = L[v] - X$ **and** $v \in T\}$
        if v_1' **and** v_3' **are connected in** T' **then** Succeeds := false
      end.

In the above procedure, the loop goes through $|V|^3$ triplets. For each triplet, it requires time $O(|N|)$ for checking the intersections and creating X, v_1', v_2' and v_3', time $O(|N||V|)$ for computing T and T', and time $O(|N||V|^2)$ for checking the connectedness of v_1' and v_3'. Hence, Fagin's algorithm is of order $O(|V|^3(|N| + |N||V| + |N||V|^2)) = O(|N||V|^5)$. As a result of this analysis, it may be claimed that our algorithm has a lower order of complexity than Fagin's. It is comparable to D'Atri's if $O(|E|) = O(|V|^{3/2})$.

Another concern is that Algorithm TEST-BETA-ACYCLICITY is based on the line graph representation of a database scheme whereas Fagin's and D'Atri's are based on the hypergraph representation. Since it takes time $O(|N||V|^2)$ to derive the line graph from a hypergraph and $O(|V|)$ is not higher than $O(|E|)$ for a connected line graph, the complexity of our algorithm remains as $O(|N||E|^2)$ even if the database scheme is given in a hypergraph representation and a transformation is required.

4.2. Creation of beta-acyclic database schemes

Algorithm TEST-BETA-ACYCLICITY may be used to redesign a database scheme by making it become beta-acyclic. This can be done by eliminating each

type of beta-cycles once detected at a step of Algorithm TEST-BETA-ACYCLICITY. For instance, pure beta-cyclic subschemes can be eliminated from a database scheme by triangulating its line graph through the addition of an attribute to two non-neighboring relation schemes in the cycle. Beta-triangular subschemes can be eliminated by relocating the attributes of the relation schemes in its incompatible triangles. Lastly, to eliminate the non-pure beta-cyclic subschemes, we simply triangulate all the duplicate-reduced subgraphs. The creation process, however, is done by adding attributes to certain relation schemes intuitively. In general, addition of attributes without careful planning may have adverse effects in the sense that, though it destroys a beta-cycle for one group of relation schemes, it may create a beta-cycle for another group. One of the open problems for research is how to perform the creation process "optimally", i.e., with the minimal changes to the given relation schemes. This problem is further complicated by the fact that logical design often involves semantic properties of the relations and attributes. This algorithm will be integrated into a graphical system for logical database design [4].

References

[1] G. Ausiello, A. D'Atri and M. Moscarini, Chordality properties on graphs and minimal conceptual connections in semantic data models, *J. Comput. System Sci.* **33** (1986) 179–202.

[2] C. Beeri, R. Fagin, D. Maier and M. Yannakakis, On the desirability of acyclic database schemes, *J. ACM* **30**(3) (1983) 479–513.

[3] T.-Y. Cheung and Y. Zhu, Recognizing acyclic database schemes in a distributed environment, Tech. Report TR-87-14, Dept. of Computer Science, Univ. of Ottawa, 1987.

[4] T.-Y. Cheung and K. Saleh, An interactive graphical system for logical database design, Tech. Report TR-88-20, Dept. of Computer Science, Univ. of Ottawa, 1988.

[5] A. D'Atri and M. Moscarini, Recognition algorithms and design methodologies for acyclic database schemes, *Adv. Comput. Res.* **3** (1986) 43–67.

[6] R. Fagin, Degrees of acyclicity for hypergraphs and relational database schemes, *J. ACM* **30**(3) (1983) 514–550.

[7] D. Maier, *The Theory of Relational Databases* (Computer Science Press, Rockville, MD, 1983).

[8] D.J. Rose, R.E. Tarjan and G.S. Lueker, Algorithmic aspects of vertex elimination on graphs, *SIAM J. Comput.* **5**(2) (1976) 266–283.

[9] R.E. Tarjan and M. Yannakakis, Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, *SIAM J. Comput.* **13**(3) (1984) 566–579.

[10] M. Yannakakis, Algorithms for acyclic database schemes, in: *Proc. Internat. Conf. on Very Large Data Bases*, Cannes, France (1981) 82–94.

[11] Y. Zhu, Line graph of gamma-acyclic database schemes and its recognition algorithm, in: *Proc. Internat. Conf. on Very Large Data Bases*, Singapore (1984) 218–221.